# Measurement and Enhancement of Cloud-based Online Gaming Systems

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Sai Ram Kowshik Vattipally

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Dr. Haiyang Wang

July 2017

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Dr.Haiyang Wang for the enormous support during my Masters study and research. This thesis would not have been possible without his guidance and help. He is kind, encouraging and genuinely cares about the well-being of his students. I could not have imagined having a better advisor for my time at UMD.

I would like to thank Dr.Douglas Dunham and Dr.Imran Hayee for serving on my thesis committee. I am grateful to all the faculty members for teaching some amazing courses and staff of the Computer Science Department for their timely help.

I owe a great debt of gratitude to Jil Pavagadhi and Sandeep Vuppula who stood by me in rain and shine and motivated me to work hard. I would also like to thank my brother, Chandra Kashyap Vattipally for his constructive criticism and taking the time to proofread this document.

## Dedication

I would like to dedicate this thesis to my parents, Venkateswara Rao Vattipally and Vani Vattipally, and to my brother, Chandra Kashyap Vattipally for their endless love and everlasting support. I feel blessed to have a family who believes in me and supports the decisions I make in life.

Abstract

Cloud gaming refers to the technologies that offload parts of game software from traditional game consoles to powerful and elastic cloud infrastructure. This design effectively shifts the game system requirements as well as the necessary computational workload to remote cloud platforms and thus has been attracting an increasing amount of attention from both service providers and end users. We have seen commercial cloud gaming systems such as Gaikai and OnLive being available in the market with a considerable user base. However, the pitfalls as well as the design challenges of cloud gaming still remains largely unclear for the general public.

In this thesis, we take an initial step towards the understanding of cloud gaming performance in virtualized environments. Different from existing cloud gaming implementations, we migrate the entire gaming system into a fully virtualized local cloud environment and compare its performance to a pure datacenter-based deployment. From our experiments, we find that the virtualized environment will significantly affect the latency as well as the frame rate of the game. As a result, we find the tested games running much slower on the cloud than on their non-virtualized counterpart. In particular, latency increases by about 176 ms and the frame rate decreases by almost 40% in our test platform. To make the matters worse, the resource utilization on the server also increases due to cloud virtualization. To address these issues, we propose a shared-memory-based enhancement to reduce the overhead of games running on virtualized environment. Our evaluation indicates that our approach can successfully reduce the cloud gaming latency by about 6% without noticeable increase of system's CPU and memory utilization.

# Contents

**List of Figures** 1

**1 Introduction** 3

**2 Background** 5

2.1 Background . . . . . 5

2.1.1 Video Games And Online gaming . . . . . 5

2.1.2 Cloud Computing And Virtual Machines . . . . . 6

2.1.3 Cloud Gaming . . . . . 9

2.2 Related Work . . . . . 14

**3 Cloud Gaming Systems** 17

3.1 GamingAnywhere . . . . . 17

3.2 Games . . . . . 19

**4 Performance And Resource Consumption In Cloud Gaming Systems** 23

4.1 Measurement Of Latency . . . . . 23

4.2 Measurement Of Resource Utilization . . . . . 26

4.2.1 CPU Usage . . . . . 28

4.2.2 RAM Usage . . . . . 31

4.2.3 Power Consumption . . . . . 33
4.3 Performance Analysis . . . . . 36
4.3.1 Latency . . . . . 36
4.3.2 Frame Rate . . . . . 39
4.4 Further Discussions . . . . . 41

**5 Optimization Of Cloud gaming Systems** **42**
5.1 Shared Memory Implementation . . . . . 43
5.2 Evaluation . . . . . 46

**6 Conclusions** **49**

# List of Figures


2.1 Type 1 virtualization . . . . . 8
2.2 Type 2 virtualization . . . . . 9
2.3 Working of cloud gaming . . . . . 12
2.4 Architecture of cloud gaming system . . . . . 14

3.1 Instance of GamingAnywhere running on a sever and thin clients . . . 19
3.2 Red Eclipse screenshot . . . . . 20
3.3 Asphalt 8: Airborne screenshot . . . . . 21
3.4 0 A.D. screenshot . . . . . 22

4.1 Screenshot of the latency measurment application . . . . . 27
4.2 CPU consumption in Red Eclipse . . . . . 29
4.3 CPU consumption in Asphalt 8: Airborne . . . . . 29
4.4 CPU consumption in 0 A.D. . . . . 30
4.5 RAM usage in Red Eclipse . . . . . 31
4.6 RAM usage in Asphalt 8: Airborne . . . . . 32
4.7 RAM usage in 0 A.D. . . . . 32
4.8 Power consumption in Red Eclipse . . . . . 34
4.9 Power Consumption in Asphalt 8: Airborne . . . . . 34
4.10 Power Consumption in 0 A.D. . . . . 35

4.11 Latency in Red Eclipse . . . . . 37
4.12 Latency in Asphalt 8: Airborne . . . . . 37
4.13 Latency in 0 A.D. . . . . 38
4.14 Frame Rate in Red Eclipse . . . . . 39
4.15 Frame Rate in Asphalt 8: Airborne . . . . . 40
4.16 Frame Rate in 0 A.D. . . . . 40

5.1 Host And Guest Memory Copy . . . . . 43
5.2 Memory Sharing Architecture . . . . . 45
5.3 Latency Comparison . . . . . 46
5.4 Frame Rate Comparison . . . . . 47

# 1 Introduction

Computer games have been a source of entertainment since early stages of computers. The number of people who play these games is estimated to be in billions today. The gaming market is expected to reach $108.9 Billion by 2017 [11]. Cloud gaming is a special form of online gaming using cloud servers. Unlike traditional online games, the game is not downloaded to the user's device but instead is hosted on a server and the game's audio and video are streamed to the gamer's device similar to that of a video stream. The gamer's input is then sent to the server and the game is rendered on the server. This moves the computational complexity away from the gamer's device. The use of cloud computing in gaming offers many advantages such as scalability, reliability and reduction of cost to gamers as well as game studios[22, 20].

Since its first commercial version by OnLive in 2009, cloud gaming has been adopted by a lot of big players in the industry in one form or the other. But the cloud gaming industry is not growing as fast as it was anticipated. The main reason for this can be attributed to QoE(Quality of Experience) in cloud based gaming systems not being up to user's expectations. Services such as PlayStation Now, Xbox One have gained popularity in recent years. But still, the network conditions are affecting the quality of game adversely. This problem is visible especially in high-end games with a large processing latency which forms a major part of all the games played today.

This in part can be attributed to the high latency in cloud gaming systems compared to regular games. For example, in a first person shooter game, a gamer has

to respond to the game in a fraction of the second. One of the reasons for the failure of the first commercial cloud gaming system, OnLive, was attributed to use of a separate physical machine for each concurrent game player[12]. Virtualizing the machine decreases operating costs for a cloud gaming service providers. No studies till today, to the best of my knowledge, proposed optimization techniques for cloud game running on VMs.

We hypothesize that by improving latency, we can improve the QoE for the user. In our work, we investigate problems associated with cloud gaming systems from a network point of view and develop methods to help improve the latency of cloud gaming systems. Current research would help billions of gamers worldwide have a better Quality of Experience while playing games by reducing the latency one might experience while using a cloud gaming platform. Also, this work will help the research community to recognize the problems associated with commercial cloud gaming systems and provides a foundation for future work.

The program developed for measuring latency as part of current work is simple to use and can be used with any cloud gaming system in general, irrespective of it being open-sourced or close-sourced. This provides a common evaluation environment for comparing various existing cloud gaming platforms. Also, the results of this study can be used to identify the areas where the current system can be improved and can open new research areas in cloud gaming systems.

# 2 Background

## 2.1 Background

### 2.1.1 Video Games And Online gaming

A video game is a computer or microprocessor controlled game. Video games create virtual situations where the game play takes place. There are many types of games. Some games simulate traditional game play methods such as cards or dice, while others might simulate a real life situation or a fantastical one. *Pong*, released in 1972 which is a 2D simulation of table tennis is considered to be the first commercial video game[23].

Initially, video games were played on dedicated machines whose hardware was designed exclusively for running games. Then, game manufacturers started porting the games to run on personal computers and mobile devices. After the advent of the Internet, online games were offered. An online game is a video game that is played either partially or completely through the Internet or any other type of computer network. Online games offered the advantage of multiplayer gaming where a gamer could compete with another gamer. Online games literally made the whole world one's playground. Players all over the world can compete with each other in the game.

Today many major companies are in the field of gaming. Activision Blizzard, Valve Corporation, Ubisoft, Electronic Arts, Rockstar Games etc. are some of the

popular game studios. Sony with its PlayStation consoles, Microsoft with its Xbox consoles and Nintendo with its Wii consoles are some of the big players in the gaming consoles industry. Also, many other major industrial giants are either directly or indirectly involved with the gaming industry. The video game market is expected to generate a revenue of $108.9 Billion worldwide in 2017 [11] and has a user base of more than 2.2 billion gamers. It is interesting to note that mobile gaming(tablets and smart phones) alone take up 42% of this market.

There are various genres of video games. These include action, adventure, role-playing, simulation, strategy, sports etc. Each genre represents a variety of games that may be further classified into sub genres. There may be games that can represent multiple genres. Each genre of game has specific characteristics in terms of response time as perceived by the user. Some games can tolerate higher latency while others need to respond fast to user inputs for a good QoE(Quality of Experience). Shea et al.[25] provides delay threshold various types of games are able to tolerate and still provide a reasonable experience to gamers.

| **Game Type** | **Perspective** | **Delay Threshold** |
|---|---|---|
| First Person Shooter(FPS) | First Person | 100 ms |
| Role Playing Game(RPG) | Third-Person | 500 ms |
| Real Time Strategy(RTS) | Omnipresent | 1000 ms |

### 2.1.2 Cloud Computing And Virtual Machines

Cloud computing is the utilization of remote resources for computational tasks instead of local resources using a distribution network, usually the web. It is also known as on-demand computing because resources, data or information are provided on demand. Cloud computing is location independent, device independent, reliable, secure, easier to manage, environment-friendly and has a lower initial investment

when compared to traditional computing [2]. The increase in the number of users on the web and improving speeds of Internet connections has greatly contributed to the widespread use of cloud computing. It is one of the popular and fast-growing techniques in the field of information technology today.

Many enterprises are moving their business to the cloud day by day. This is because of the many advantages of cloud computing versus hosting their own machines. Organizations do not need to worry about maintaining their infrastructure for round the clock availability or bearing additional costs for having redundancy in case something fails. Also, infrastructure can be scaled up dynamically on demand without having to worry about costs of buying new hardware in case of increased business needs, or it can be scaled down without any concern for underused hardware. The clients are charged only for the time for which they have used the resources by the cloud service provider, thus resulting in low cost. The data on the cloud can be accessed from anywhere in the world. Also, cloud providers provide service level agreements which guarantee a minimum level of service quality, thus ensuring a consistent service.

A virtual machine(VM) can be considered as a base unit in cloud computing. It can be understood as an application environment that is installed as a software but imitates a dedicated hardware. VMs are used to simulate separate physical computers on the hardware of a single computer. A hypervisor is a low-level program that acts an intermediate between the host and the guest. It is a thin software that either runs on an operating system or as a firmware directly over hardware. Virtualization can be broadly classified into two types:

- Type 1 or Hardware Virtual Machine(HVM) in which hypervisor directly runs over the hardware and hence any operating system can be run on an HVM

without any modifications. Hence, this is called hardware virtualization. Type 1 virtualization generally provides better performance as compared to type 2 virtualization since the hypervisor can access the hardware directly but this requires native hardware support to support a hypervisor directly.

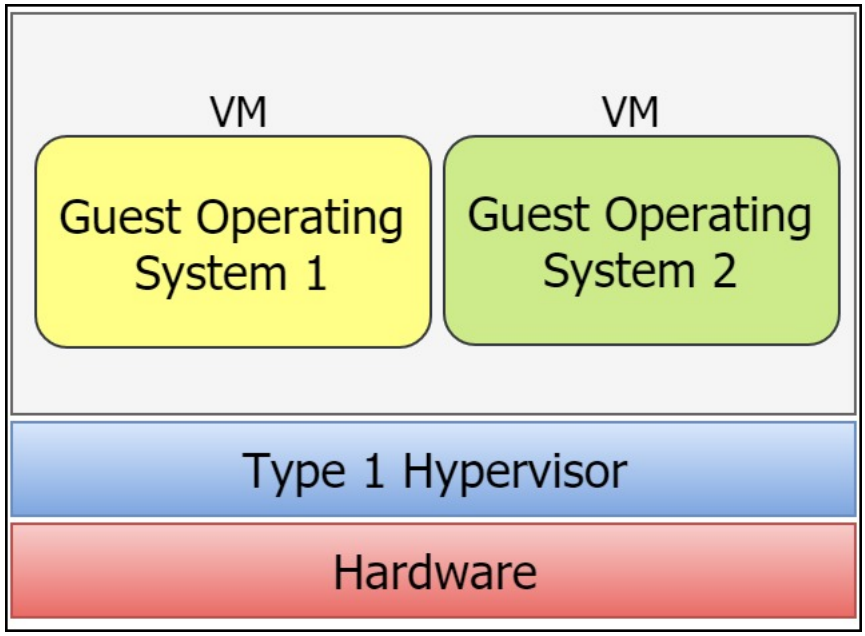


Figure 2.1: Type 1 virtualization

- Type 2 or Para-virtualization(PV) in which a hypervisor operates over an existing operating system. Here the hypervisor translates the calls from guest operating system to calls the host operating system supports. Hence, this is called software virtualization. The hypervisor also controls virtual memory and I/O management. Type 2 virtualization environment is easier to install as one can install a virtualization software over an existing operating system and start creating virtual machines. Type 2 virtualization is generally less efficient than type 1 virtualization but supports a broad range of I/O devices that the native operating system supports.

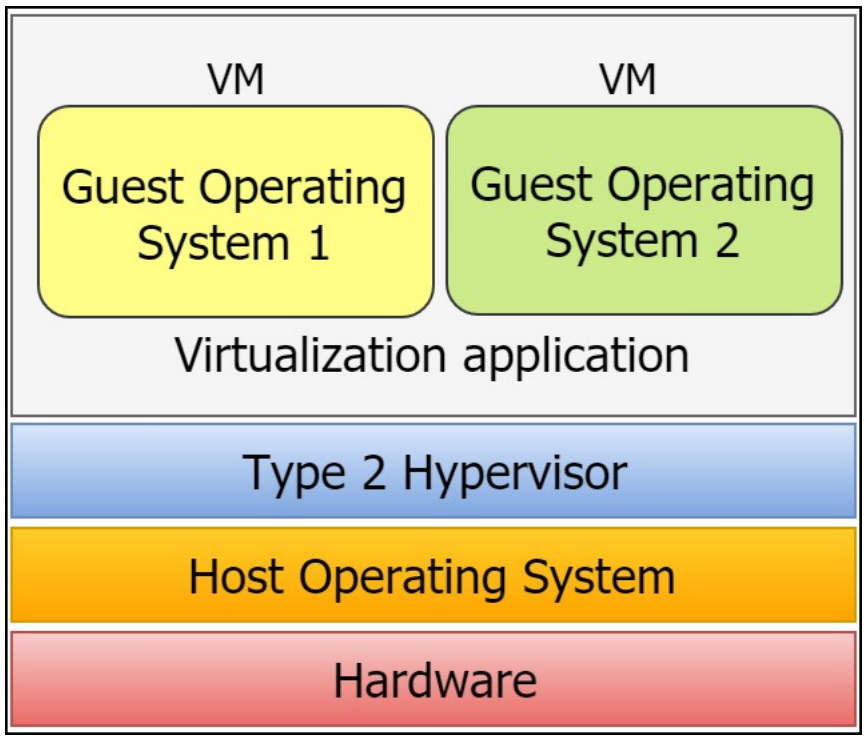


Figure 2.2: Type 2 virtualization

### 2.1.3 Cloud Gaming

Cloud gaming is the utilization of cloud services for online gaming. Cloud gaming employs a client-server architecture. The game is hosted on a cloud server whereas the users can use thin clients to play these games via a network. A thin client is a lightweight computer that is capable of establishing a remote connection to a server. All the important computational activities such as game rendering and AI are done on the server. Hence the computational complexity is shifted to the server and the client has to merely display the game stream and send the game control inputs to the server. This enables any device which is typically able to play a video stream to work as a gaming client. The game is streamed to the client as video and audio streams, and the game control actions received as input by the users are received through the network and rendered on the server to generate the corresponding response.

Because of the wide-spread cloud services and data centers, cloud gaming, which was once thought of as an academic experiment has turned into reality. Today many major companies such as Sony, Microsoft and Nvidia have entered the market of cloud gaming.

It provides the advantage of reducing the cost required for playing high-end games by reducing hardware costs since the gamer can use a thin client which he already owns, instead of buying a new gaming console that supports the game he intends to play. Also, a game developer can support many platforms using same or similar applications and can have higher revenues as compared to traditional games.

Cloud gaming systems provide advantages to both game players as well as game developers. Game players have the following advantages:

- No hardware upgrades required: Cloud gaming demands minimal hardware requirements as compared to running a game locally. A game player need not upgrade his hardware to stay in tune with demands of new games that constantly demand higher hardware requirement minimums.

- Rent new games, not buy them: Depending on the business model of the cloud gaming provider, services are provided at a monthly fee for all the games available with the provider or a monthly fee per individual game which the user wants to play. This means that a game player need not buy new games as they are released but instead rent them.

- Multiple device support - Move your game: Any device that can run a gaming client software can be used to play cloud games. This includes a wide array of devices such as smart phones, tablets, laptops, personal computers, game consoles and TVs. This means that a user can play a game on one device, save it to the cloud and resume the game on another device at a later time.

- Heterogeneous platform support: Some games today are only supported on certain platforms. For example, Unchartered[1] is only available on Sony PlayStation consoles. Forza Horizon[2], Halo[3], Gears of War[4] are only available on Microsoft Xbox console. Similarly, there are certain games that only support Windows OS. With cloud gaming, any game can be played on any platform as a server running on any platform can stream to a client on any platform.

- No downloads or updates: Game Players need not download installation files that are typically a few GB. This gives the player the ability to play any game instantly. Also, the gaming provider typically takes care of updating the game or applying patches, so less down time.

Game studios have the following advantages:

- Develop a single version of the game for all platforms: A single version of the game can be developed for users on any platform reducing development costs.

- Piracy control: Games are run on the server while the client only sees the output from the game. The users don't have access to source code of the game. Hence this prevents piracy of the game.

- Easy deployment: New games can be easily deployed with a single release to all the users on all the platforms which also translates to savings.

- Easy maintenance and updates: Game updates and patches can be done quickly without the user having to do anything.

[1] https://www.unchartedthegame.com/
[2] https://forzamotorsport.net/en-US/games/fh3/
[3] https://www.halowaypoint.com/
[4] https://gearsofwar.com/

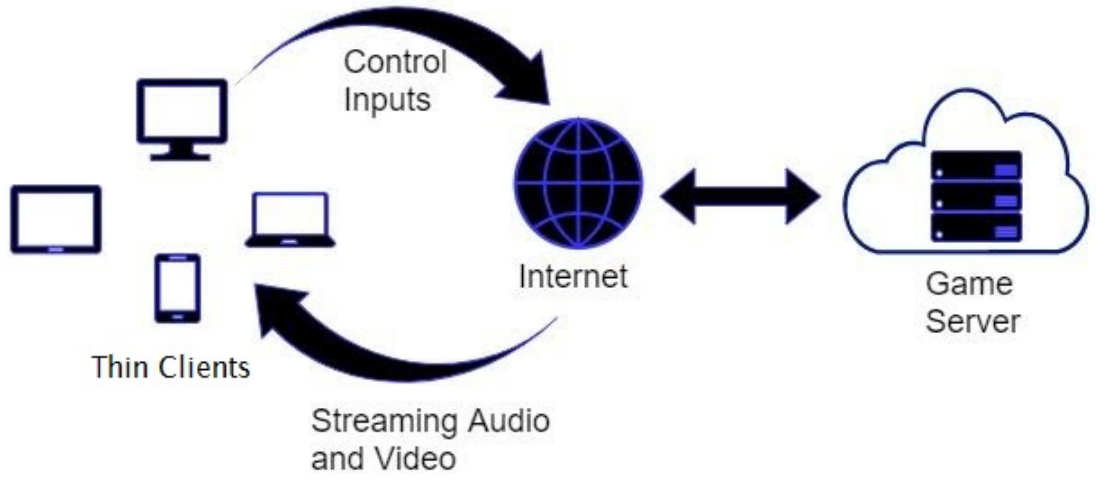


Figure 2.3: Working of cloud gaming

In addition, these cloud gaming platforms can also be used in fields other than entertainment such as visualization in scientific simulation. For example, high-end graphics on cloud gaming servers can be exploited to run applications such as computer-assisted Design (CAD), computer animation, information visualization, surface rendering and volume rendering.

But these systems face many problems. Games are real time systems unlike videos, hence there is no concept of buffering and acquiring the data in advance. Also depending on the network latency, the gamer may experience delays between host actions and the response of this system. Delay to a certain extent may not affect the QoE. This delay is called delay tolerance. The delay tolerance is different for various types of games. For example, first person shooting games have less delay tolerance while strategy games have higher delay tolerance[8, 25]. One other issue is the type of encoding used for compressing data before sending it on the network as compressing and uncompressing may take some time and add to the delay. Also sometimes it

might be necessary for cloud gaming systems to reduce the resolution or frame rate of the video to transfer data fast. This decreases the richness of gaming experience to the gamer.

Traditionally games were played using keyboard and mouse, but most of the thin clients used for cloud gaming such as mobiles and tablets have touch interfaces. Most games today are developed with a PC or console in mind and are not optimized for control using touch screens. Mapping the touch inputs to keyboard presses or mouse movements is another challenging task. Optimizing servers that are used for running games is very different from optimizing normal servers. We need to have a balance between different types of games that run on a single physical server that utilizes a different amount of resources[14]. For example, we can run a higher number of games that require low resources or a lower number of games that require high resources. Also, different games require different types of resources like GPU, CPU or memory. Optimizing the game according to the client's environment is another challenge. We have to optimize the game according to the network connection, latency, screen size and resolution. Hence it is very important to measure various factors that affect the performance of cloud gaming system. Many approaches were proposed and used for measuring the optimality of such systems[28][9].

A cloud game implemented in VM on a server is prone to further problems. In addition to network issues, it is usually found to be slower than games running on independent systems. Measuring the differences between similar instances of games running on a local machine and a VM on cloud server gives us essential information about the problems cloud games face on VMs.

## 2.2 Related Work

Cloud Gaming has been a hot topic among the research community recently. Chen et al.[8] published a comprehensive study highlighting various aspects and research opportunities involved in developing a cloud gaming system.

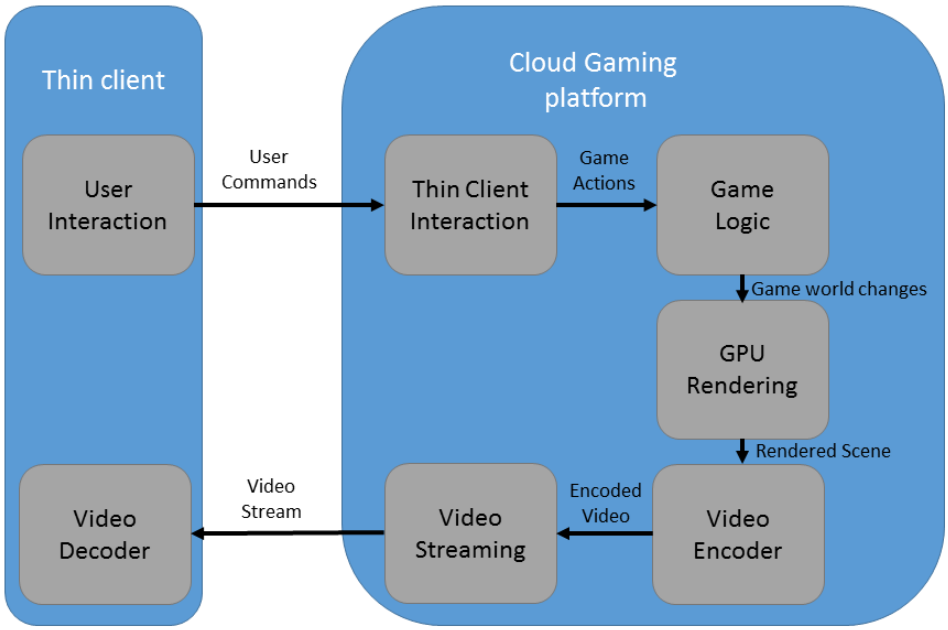


Figure 2.4: Architecture of cloud gaming system

A study by Hong et al.[13] compared the trade-offs between *Quality-of-experience* (QoE) of a game and hardware requirements on the server. They suggested using *server consolidation* which would enable dynamic allocation of resources among game servers. This would result in a better overall performance and lower operational cost for the service provider.

Another experiment by Zhao et al.[29] involved the thin client in processing the game along with the server.The task of running the game would be divided between

the client and server. Doing all the game related stuff on the server would increase the latency as it takes some time. So instead they propose to run certain tasks on the client. They used multiple VMs to do each task of game processing and this allocation would be dynamic. For example when a thin client has good CPU capabilities but low GPU capability, the CPU intensive tasks such as AI, Animation, Audio rendering, game encoding etc. would be done by the thin client where as the GPU intensive tasks such as Graphics and Physics simulation would be taken care by the server. They divide each part of the game processing into different parts and assign a VM to perform each task. Also, when all the VMs complete their tasks the encoding VM is woken up to integrate all these parts of the game. Also, this team for the first time virtualized GPU to create several instances of GPU VMs to parallelize GPU intensive tasks as well. This method showed good improvements in the processing times and resulted in a significantly less latency in the game. But the main problem that this system faces is that every game cannot be divided into parts. In addition, one has to have a thorough knowledge of how a game is implemented and should be able to divide the gaming tasks into various parts that can run independently. Also, it is hard to access the source code of most games in the first place as many of them are closed-sourced and proprietary. Any implementation should be commercially viable and should be applicable to a wide array of games irrespective of it being an open -source or close-sourced game.

Barboza et al.[6] proposed a VM approach to provide Gaming-on-Demand(GoD) services on the cloud platform. They proposed to use three levels of managers for the cloud, hosts and clients. The cloud manager monitors the host managers' activities and controls the provisioning of VMs for video streaming to clients. Klionsky[16] developed an architecture for image rendering and amortized graphics based leveraging cloud technology. Lou and Hwang[19] developed a peer to peer model to eliminate

video jittering in video streaming and IPTV applications. Another P2P approach supporting online gaming was developed by Suselbeck, et al[26]. Jurgelionis, et al[15] studied the impact of networking on gaming.

Studies that were done until now concentrated on optimizing a specific issue of a cloud. One such solution would be to have a much more distributed network so that the latency can be decreased by having small data centers closer to the users rather than having a huge data center far away from the users. Cong Ding et al.[10] proposed methods to allocate a server to the user dynamically in a distributed cloud computing network based on factors such as the distance of the user from the data center, current work load on the server and network conditions. They ultimately developed a middleware for cloud applications named CloudGPS for measuring these network parameters and taking a decision on allocating the optimal data center in the cloud.

Today, the field of cloud gaming is growing faster than ever before with various new players such as Liquid Sky[18], Parsec[21] and Vortex[27], and a lot of research has already been done on methods used to run games on a cloud. But the problem is that all the cloud gaming systems are closed-source and proprietary. An open-source cloud gaming system called gaming anywhere has been developed by Chun-Ying Huang and team[14] which will be used as a test bed for measuring and comparisons in the current study.

# 3 Cloud Gaming Systems

## 3.1 GamingAnywhere

GamingAnywhere is an open-source cloud gaming platform. It is a custom-developed software that employs various network protocols such as RTSP over UDP, RTSP over TCP and also its own protocols to simulate a cloud game. Firstly the screen is captured using screen hook functions on Windows or Linux. The video stream from the server to a client is sent using RTSP over UDP by default. TCP can be used instead of UDP if desired. GamingAnywhere uses its own custom protocol for sending the control events from client to a server. On detecting a control event(such as mouse clicked, mouse moved, keyboard button pressed), the event is packed into a control message and sent to the server through the control protocol.

GamingAnywhere works in two modes, Periodic mode and Event-Driven mode. In Periodic mode, the user specifies a specific window to capture. Otherwise, the whole desktop is captured by default. *ga-server-periodic* automatically tries to find the window-name specified in the configuration file, moves the game to top left and captures the video and audio generated only from that game window. In this mode, GamingAnywhere can simply be used as a Remote Desktop Control software if no window is specified and captures and streams the whole desktop. A game could simply be made to run in on the full screen and captured using Periodic mode when we couldn't use Event-Driven mode.

In Event-Driven mode, the *ga-server-event-driven* launches the game specified in

configuration file. The application is specified by using the "game-dir" (optional) and the "game-exe" (mandatory) options. Hence in this mode, each game has its own configuration file in which the directory of the game and the executable file of the game are specified.

GamingAnywhere uses IP addresses for connecting a client to the server. By default, it uses an RTSP protocol connection over UDP using port 8554 on the server. GamingAnywhere doesn't have a GUI yet. It should be launched using console commands. Also it uses a set of configurations files to specify settings such as frame rate, maximum connection speed etc. Each game or connection can be configured by modifying a specific configuration file. All the games in our experiments are run at Full HD resolution(1920 * 1080 px).

Listing 3.1: Commands to run GamingAnywhere Server

```
1 Periodic Mode
2       $ ga-server config/server.desktop.conf
3 Event-driven Mode
4       $ ga-server-event-driven config/game-name.conf
```

The server program in GamingAnywhere is responsible for Capturing the game video and audio and streaming it to the client. It uses the hooking mechanisms of the respective operating system to capture the screen and make this into an RTSP stream to send to the client.

The client in GamingAnywhere receives the data sent by the server and displays the video on the screen. Also, it captures the user actions and sends them to the server. Since there is no standard protocol that fits the need, GamingAnywhere implements its own control protocol to deliver control events from a client to a server.

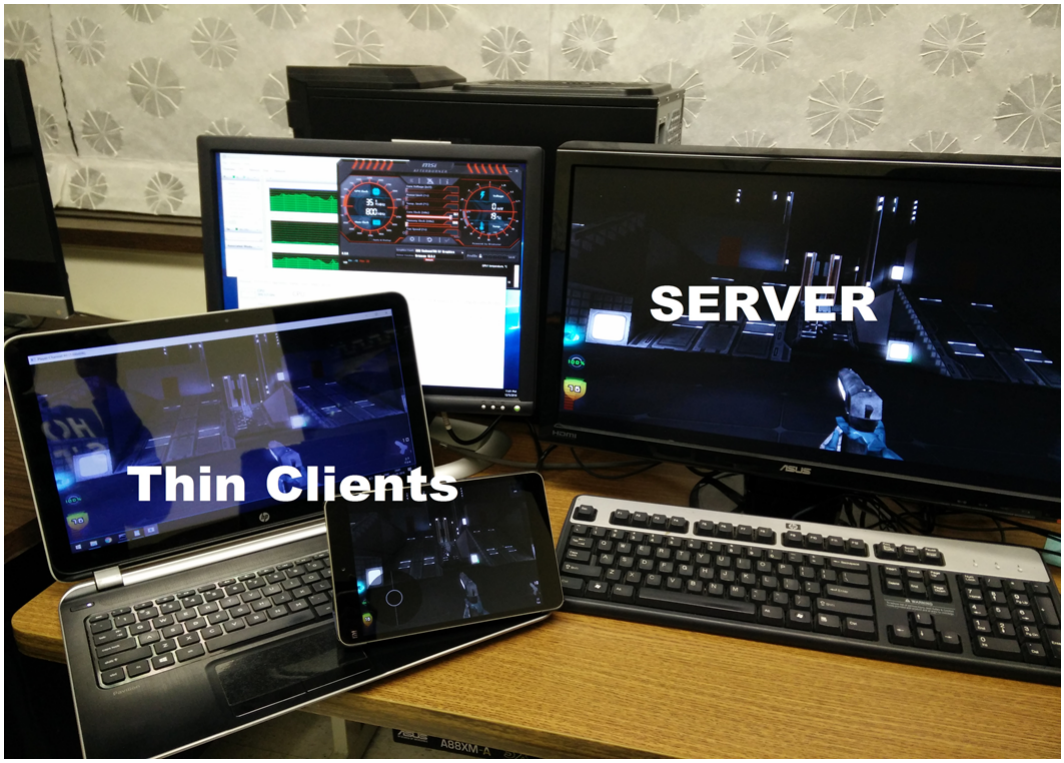


Figure 3.1: Instance of GamingAnywhere running on a sever and thin clients

Listing 3.2: Command to run GamingAnywhere Client

```
1 #Running client
2 $ ga-client config/client.rel.conf rtsp:/
3         /<server ip address>:8554/desktop
```

## 3.2 Games

As games of different genres have different characteristics, we try to pick each game from a different genre. For example, First Person Shooter(FPS) games are usually optimized to respond faster than Role Playing Games(RPG) while RPG games are optimized to provide better graphical performance. The games we use for our experiments are

1. Red Eclipse[24] : A 3D first person shooter game where the player fights against robots using various kinds of weapons in a closed arena. It is an open source game in which a player can compete with the computer or with other players in a multi-player online gaming mode. The objective of the game is to obtain maximum kills or survive for the longest depending on the kind of match chosen. It is available to download from the game website. It was built on Cube Engine 2 using SDL and OpenGL.


Figure 3.2: Red Eclipse screenshot

2. Asphalt 8: Airborne[4] : 'Asphalt 8: Airborne' is a 3D racing game developed by Gameloft as part of the Asphalt series. It is a classic car racing game where a player can race in various scenarios such as classic race, time trial, elimination etc. The goal of the game is to stand first in the race or perform various other maneuvers during the race depending on the game objective. The player can play a single player game locally or can compete against other players locally via Wi-Fi or globally via the internet. The PC game is available for download on the Windows Store for computers running Windows Operating system.


Figure 3.3: Asphalt 8: Airborne screenshot

3. 0 A.D.[1] : 0 A.D. is a real-time strategy game developed by WildFire Games. It is a historical war and economy focused game where a player has to build his civilization by gathering various kinds of resources, training an army for combat against other civilizations and technology research. The player has to advance through multiple phases to unlock new units, buildings and technologies. The objective of the game is to build a prosperous civilization and compete against other civilizations.


Figure 3.4: 0 A.D. screenshot

# 4 Performance And Resource Consumption In Cloud Gaming Systems

## 4.1 Measurement Of Latency

Since cloud gaming systems are the results of recent developments, there was no software that was available out of the box to measure latency. Hence we chose to develop our own program that could be used for measuring latency in cloud gaming systems. We choose to develop this program using C Sharp(C#). This is because the libraries available with C# provided better performance as compared to other options since they interacted well with native Windows operating system which we were using for running our experiments.

Initially, we experimented with a mechanism where we took the screenshot of the whole game screen at a random time after we send an input to the game and examine the presence of reaction for that particular input to determine the latency, similar to method used in Chen et al.[7]. This is to be repeated multiple times to find a list of times in which there was a reaction and a list of time in which there was no reaction. The border between these two lists is supposed to be the latency. But we found that the time taken for capturing the screenshot was about 40-60 ms and was much larger than what would prove to be a reasonable time. This was problematic since every millisecond counts while measuring latency and this kind of delay is unacceptable for

our measurements.

Hence, we use the method presented in Lampe et al.[17]. Initially, we choose an input that would generate an expected output on the game screen. We also choose a rectangular reaction area on the screen where we expect to see a reaction for our input. This reaction can be understood as a gunshot being fired at a mouse click or car moving right as a result of pressing the right arrow key on the keyboard. We simulate the said input using our software and record the time as $t_0$. Then, we capture the color values of the pixels in the reaction region from the front buffer and measure the current color average value as $c_{init}$. Then this process is repeated each time updating the current average $c_{curr}$ and the time is recorded as $t_1$. This process is repeated until the absolute difference in color average is greater than a predefined threshold $\delta$ (i.e. $|c_{curr} - c_{init}| > \delta$). When this condition becomes true, the value of $t_1$ is saved. $t_1$ is considered as the time at which the reaction to the input is observed on the game screen. Then latency would be calculated as

$$Latency = (t_1 - t_0)$$

Multiple reading are taken at various stages of the game and the mean of the results is calculated.

The algorithm for measuring latency can be summarized using the following Pseudo code. The screen shot of the application is shown in 4.1.

Listing 4.1: C# Pseudo code for measuring latency

```
1 //Select a screen region using top left and bottom right
2 //screen co-ordinates where the measurement should be done
3 int topleftX = 0;
4 int topleftY = 0;
```

```
5  int bottomRightX = 1920;
6  int bottomRightY = 1080;
7
8  //Select a color change threshold
9  int threshold = 500000;
10 int colorAvgChange = 0;
11 int numberOfPixels = (bottomRightX - topLeftX) *
12                       (bottomRightY - topLeftY);
13 //Register a game input event
14 if(keyboardInput)
15 inputEvent = chosenKey;
16 else
17 inputEvent = chosenMouseClick;
18
19 //start a timer
20 Time t1 = Timer.start();;
21 simulate(inputEvent);
22 C_init = computeColorAverage(topLeftX, topLeftY,
23                                bottomRightX, bottomRightY);
24 C_curr = C_init;
25 while(abs(C_init - c_curr)< threshold)
26 {
27     C_curr = computeColorAverage(topLeftX, topLeftY,
28                                bottomRightX, bottomRightY);
29 }
```

```
30
31 Time t2 = Timer.stop();
32
33 latency = t2 - t1;
34
35 double computeColorAverage(int topLeftX, int topLeftY,
36                     int bottomRightX, int bottomRightY)
37 {
38     double colorSum = 0;
39     for(int i = topleftX, i <= bottomRightX; i++)
40         {
41             for(int j = topleftY; j <= bottomRightY; j++)
42             {
43             colorSum += Color(pixel[i,j]);
44             }
45         }
46     double avgColor = (colorSum/numberOfPixels)
47     return avgColor;
48 }
```

## 4.2 Measurement Of Resource Utilization

We measure the CPU usage, RAM usage and power consumption in following scenarios:

- Scenario 1: Game ran locally (on a bare metal computer).

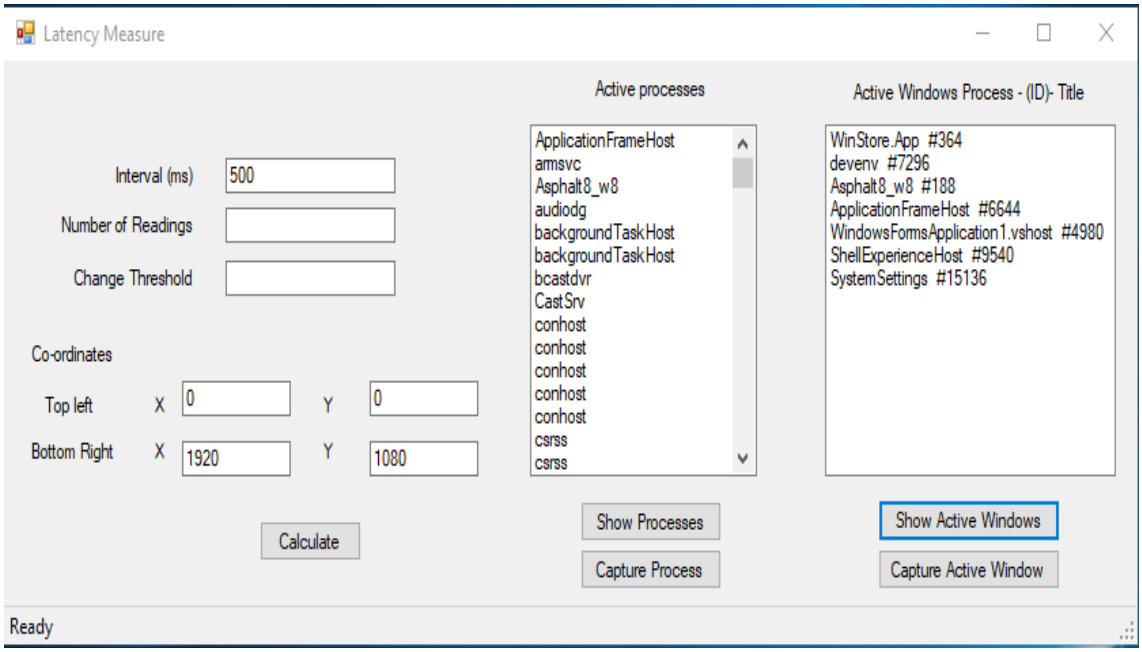


Figure 4.1: Screenshot of the latency measurment application

- Scenario 2: Game streamed using a cloud gaming system.
- Scenario 3: Game streamed using a cloud gaming system ran inside VM.

We run all our experiments on systems with following configuration:

| | |
|---|---|
| **CPU** | AMD A10-7860K Radeon R7 4 Core 3.60GHz |
| **RAM** | 8.00 GB |
| **GPU** | AMD Radeon R7 1 GB |
| **Storage** | 128 GB SSD |
| **Operating System** | Windows 10 Pro |

Both client and server in our experiments have the same configuration.

CPU usage and RAM usage are measured using Resource Monitor provided in the Windows Operating system on the host. Power consumption was measured using an Electricity usage monitor. The VM is provisioned using Windows 10 Pro operating system on VMWare Workstation Pro 12.5.7 with the maximum amount of resources

that can be allocated to the VM by the software. This provisions a VM using para virtualization. Our intuition here is that when a bare metal computer is virtualized, some of the resources on the system are used for virtualization. Hence by taking this approach, we get an estimate of the resources consumed by the virtualization software along with the cloud gaming software in our experiments.

The difference between scenario 1 and scenario 2 are caused by the additional overhead for running the cloud gaming system where as the difference between scenario 2 and scenario 3 is caused due to virtualization. Hence the additional resources used in scenario 2 over scenario 1 are used by the cloud gaming system for various activities such as receiving packets containing game inputs, unpacking them and sending these inputs to the game, capturing game video, encoding it, packaging them into UDP packets and transmitting these packets onto the internet. Similarly, the additional resources used in scenario 3 over scenario 2 are used by the virtualization software for provisioning the VM with resources, acting as an agent between the host and guest for communications etc.

We measure the resources consumed by each game when it is running in various scenarios. The results were as follows.

### 4.2.1 CPU Usage

| Game | Bare metal game | Cloud Game | Cloud Game in VM |
|---|---|---|---|
| **Red Eclipse** | 29.38% | 81% | 91.13% |
| **Asphalt 8: Airborne** | 21.96% | 62.71% | 93.34% |
| **0 A.D.** | 22.16% | 71.38% | 85.61% |

Figures 4.2, 4.3 and 4.4 represent the CPU usage in various scenarios.

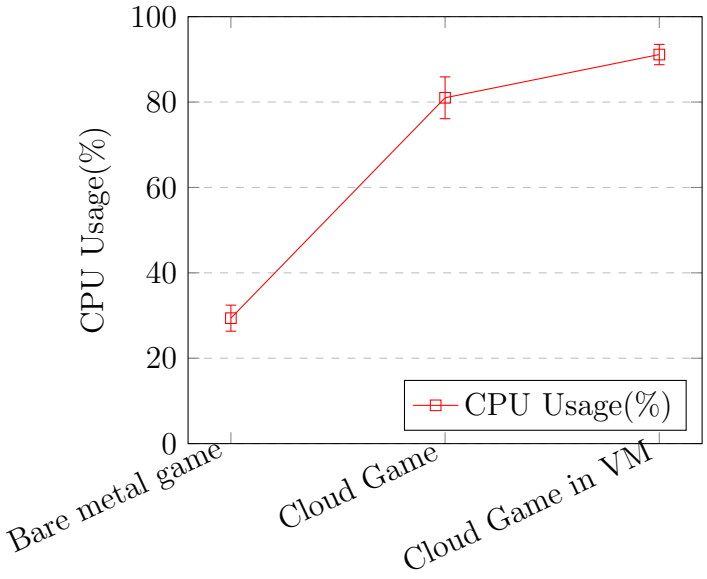


Figure 4.2: CPU consumption in Red Eclipse

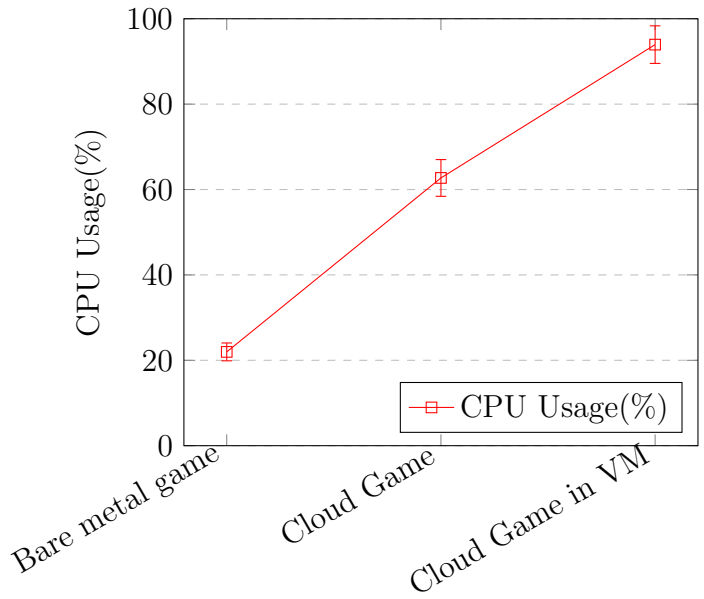


Figure 4.3: CPU consumption in Asphalt 8: Airborne

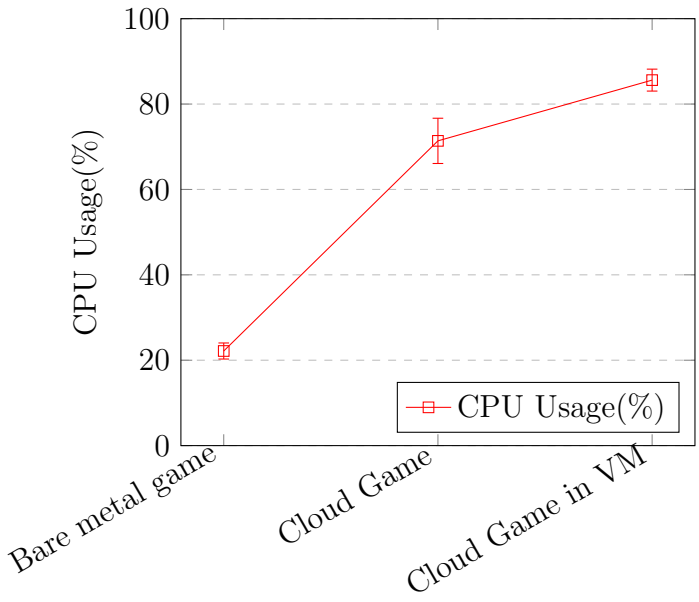


Figure 4.4: CPU consumption in 0 A.D.

From the above results, we observe that the cloud gaming software consumes about 47.19% of the CPU on an average over what a game running on a bare metal system would consume. As expected this CPU overhead is due to the high computational requirements of a cloud gaming systems. Then, we also observe an average increase of 18.53% in CPU consumption due to virtualization. This is due to various tasks the virtualization software is doing coordinating between the host and guest OS.

### 4.2.2 RAM Usage

| Game | Bare metal game | Cloud Game | Cloud Game in VM |
|---|---|---|---|
| **Red Eclipse** | 2028.46 MB | 2066.94 MB | 5766.76 MB |
| **Asphalt 8: Airborne** | 1904.20 MB | 1912 MB | 5796.42 MB |
| **0 A.D.** | 2319.23 MB | 2601.90 MB | 5958.55 MB |

Figures 4.5, 4.6 and 4.7 represent the RAMusage in various scenarios.

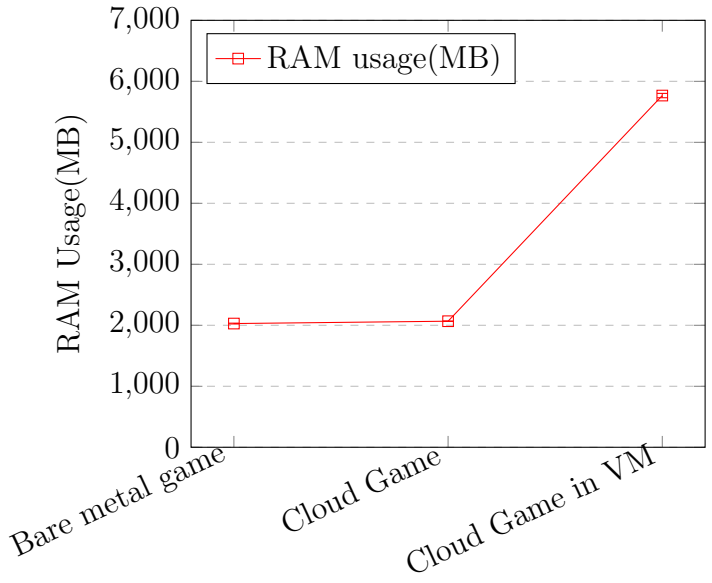


Figure 4.5: RAM usage in Red Eclipse

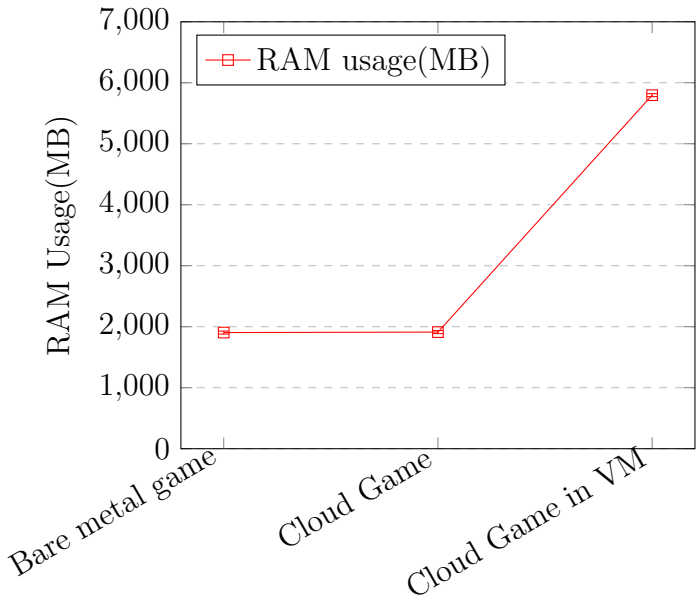


Figure 4.6: RAM usage in Asphalt 8: Airborne

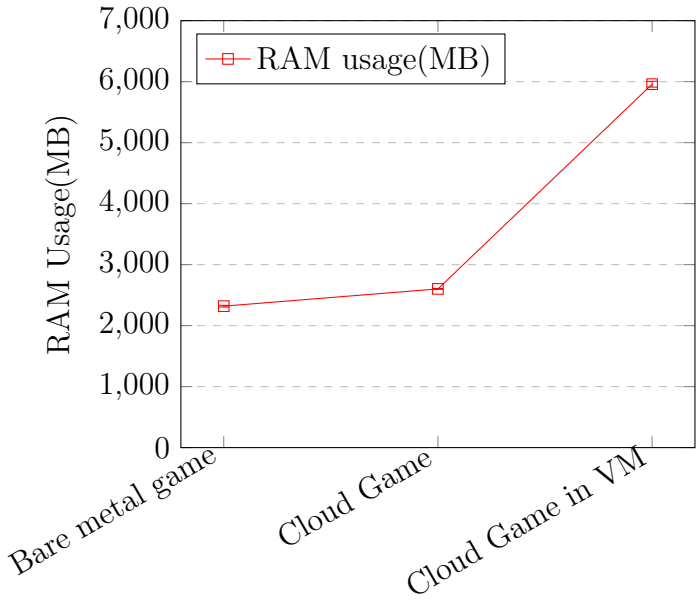


Figure 4.7: RAM usage in 0 A.D.

From the above results, we observe that the cloud gaming software consumes about 109.65 MB of the RAM on an average over what a game running on a bare metal system would consume. This is not a lot of increase in RAM usage. In fact, in Red Eclipse and Asphalt 8, this increase was only 38.48 MB and 7.8 MB respectively. This means that a cloud gaming system does not require a lot of additional memory. This is because these systems are real-time systems which have to process and send out the data immediately and there is no concept of buffering and storing the data in memory. Then, we observe an average increase of 3646.96 MB in RAM usage due to virtualization. This is due to fact that the virtualization software is hosting a guest OS which requires a relatively high amount of RAM to run.

### 4.2.3 Power Consumption

| Game | Bare metal game | Cloud Game | Cloud Game in VM |
|---|---|---|---|
| **Red Eclipse** | 86.05 W | 101.15 W | 102.17 W |
| **Asphalt 8: Airborne** | 71.21 W | 97.79 W | 103.3 W |
| **0 A.D.** | 85.015 W | 107.59 W | 108.08 W |

Figures 4.8, 4.9 and 4.10 represent the power consumption in various scenarios.

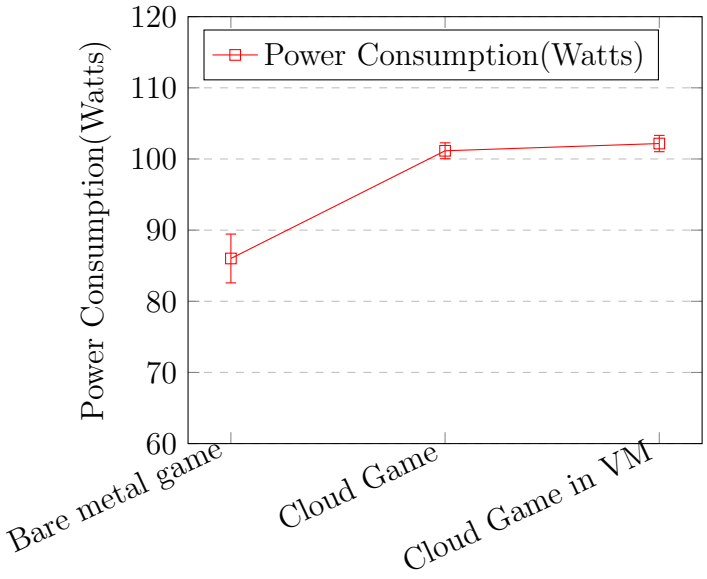


Figure 4.8: Power consumption in Red Eclipse

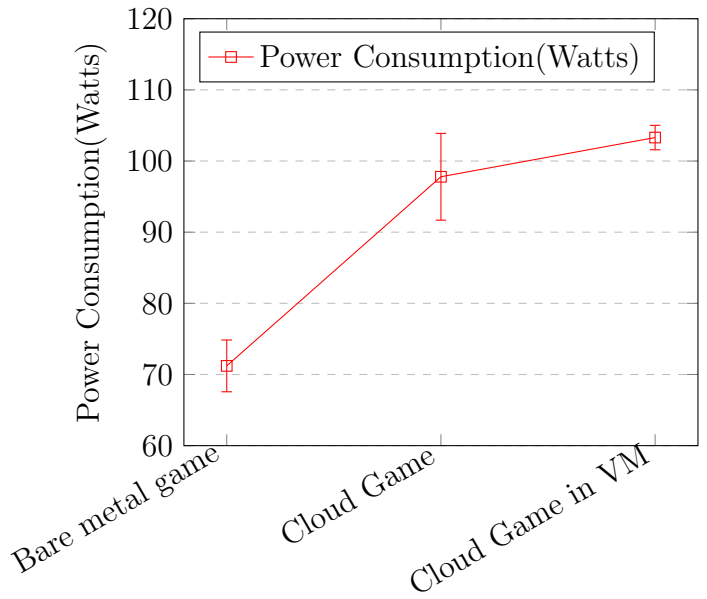


Figure 4.9: Power Consumption in Asphalt 8: Airborne

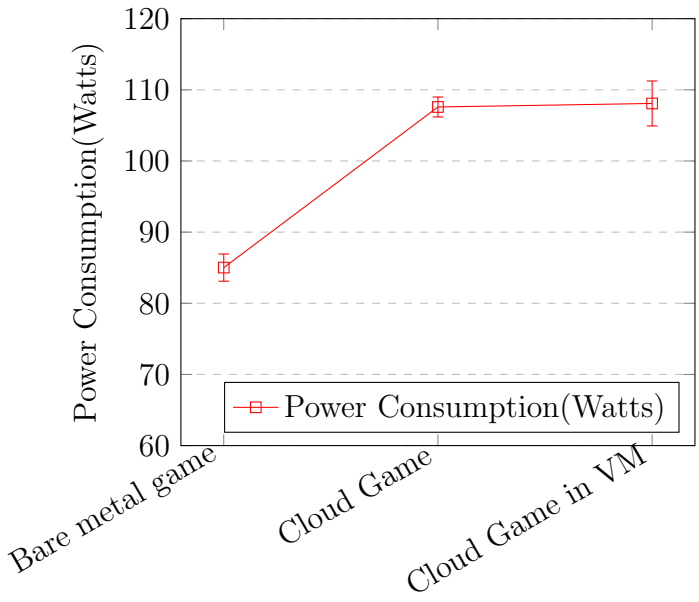


Figure 4.10: Power Consumption in 0 A.D.

From the observed power consumption values, we notice that the cloud gaming software increases the power consumption by 21.41 Watts on an average. This accounts for about 26% increase in the power consumed when compared to the power consumed by game running on a bare metal system. This increase is in line with the increased CPU utilization as CPU and GPU are the biggest consumers of power when a computer runs games. Also, we observe an average increase of 2.34 Watts in power consumption due to virtualization. Although this is not as high as compared to increase in the CPU consumption due to virtualization, we attribute this to our virtualization software not being able to fully utilize the discrete GPU available and hence a decrease in the power consumed by GPU.

## 4.3 Performance Analysis

We measure the performance of cloud gaming systems using latency and frame rate of the game in various scenarios. We measure the latency using the software we built. The frame rate is recorded using the in-game frame rate display when running on a bare metal machine in case of Red Eclipse and 0 A.D., while the frame rate is measured using the software *Fraps* when an in-game frame rate display is not available or when the game is running on a cloud-gaming system.

It is important to note that both the server and client were on the same university network which runs on a Gigabit Ethernet backbone and the latency between the two computers was always less than 1 ms. Hence, the network conditions were never a limiting factor for the performance, unlike real-world conditions where the network conditions are considered the main bottle neck. This has been done in order to measure the performance of hardware accurately.

### 4.3.1 Latency

| **Game** | **Bare metal game** | **Cloud Game** | **Cloud Game in VM** |
|---|---|---|---|
| **Red Eclipse** | 51.94 ms | 354.48 ms | 491.97 ms |
| **Asphalt 8: Airborne** | 107.21 ms | 413.16 ms | 593.08 ms |
| **0 A.D.** | 93.59 ms | 354.09 ms | 567.58 ms |

Figures 4.11, 4.12 and 4.13 represents the latency in various scenarios.

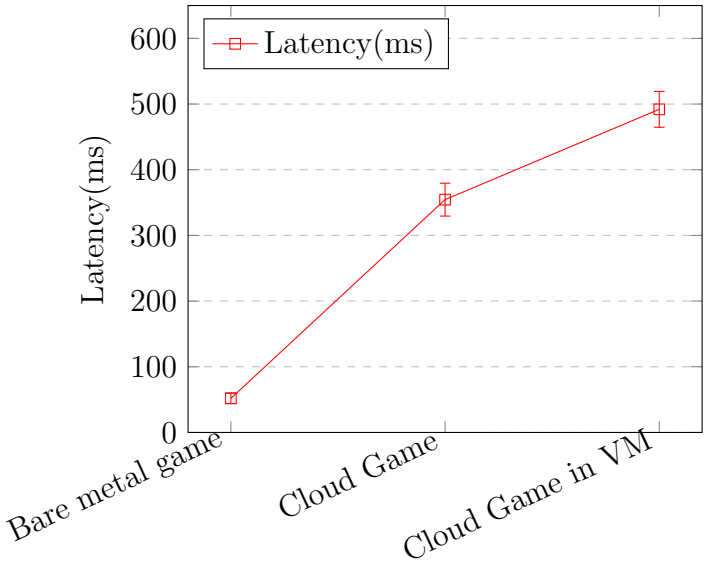


Figure 4.11: Latency in Red Eclipse

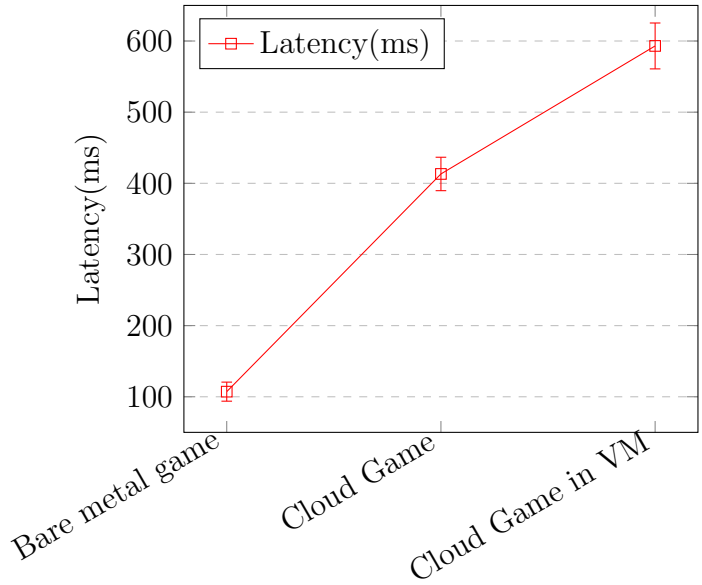


Figure 4.12: Latency in Asphalt 8: Airborne

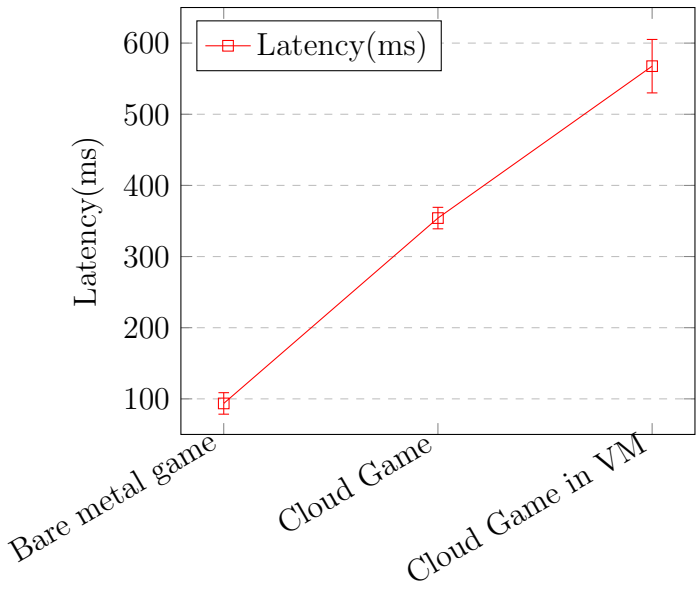


Figure 4.13: Latency in 0 A.D.

From the above results, we observe that games that have an average latency of 84.24 ms on a bare metal computer have an average latency of 373.91 ms when running on a cloud gaming system. This increase of 289.67 ms can again be attributed to various tasks a cloud gaming systems perform that include capturing game video, video compression, encoding and decoding of packets. Then, virtualization results in a further increase of 176.96 ms. This is caused due to the additional latency caused by the hypervisor in transferring the messages from guest OS to host OS and returning the messages from host OS to guest OS.

### 4.3.2 Frame Rate

| Game | Bare metal game | Cloud Game | Cloud Game in VM |
|---|---|---|---|
| **Red Eclipse** | 53.36 fps | 13.1 fps | 6.92 fps |
| **Asphalt 8: Airborne** | 30 fps | 10.52 fs | 6.80 fps |
| **0 A.D.** | 32.47 fps | 10.66 fps | 7.04 fps |

Figures 4.14, 4.15 and 4.16 represent the frame rate in various scenarios.

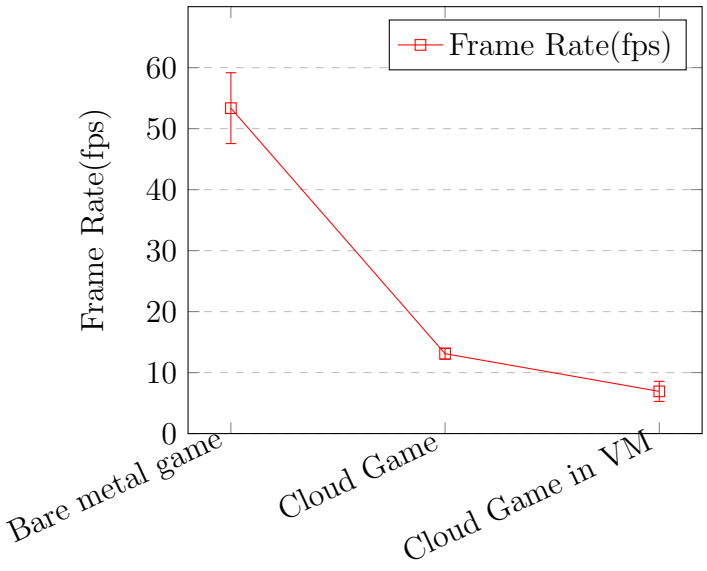


Figure 4.14: Frame Rate in Red Eclipse

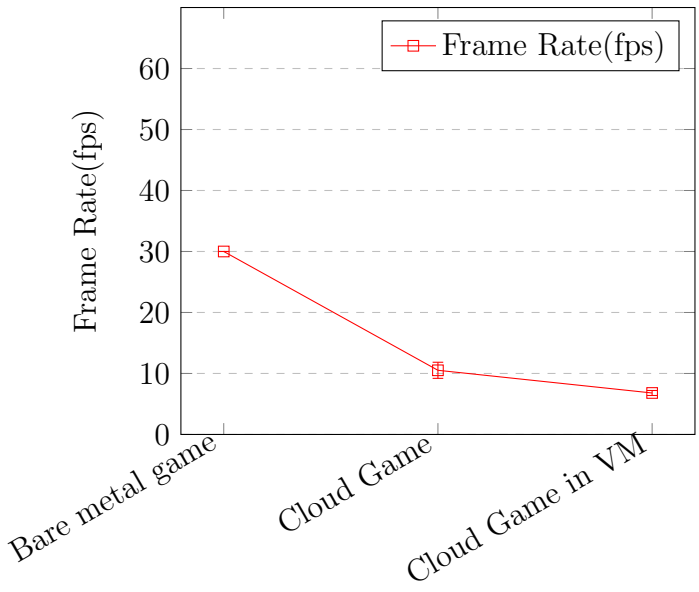


Figure 4.15: Frame Rate in Asphalt 8: Airborne

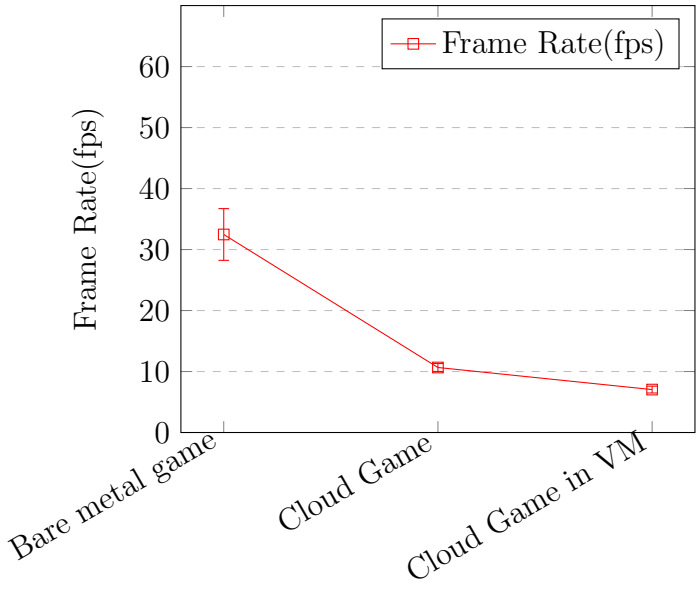


Figure 4.16: Frame Rate in 0 A.D.

From the above results, we observe that games run at an average frame rate of

38.6 fps on a bare metal computer. This drops to 11.42 fps on a cloud gaming system. The frame rate is usually directly proportional to the amount of GPU and CPU a game can utilize in a given time and hence higher utilization can generate more frames in a limited time. As the CPU utilization by the cloud gaming system increases, the actual game will not be able to get CPU cycles as frequently as it did before. Also, the GPU is now additionally used for video capture and compression and hence is not exclusively available to render the game.

Virtualizing this computer results in a further decrease to 6.92 fps. This is caused due to the additional competition for CPU utilization by the virtualization software. This can also be partially attributed to VMWare workstation's inability to completely virtualize underlying GPU on the computer.

## 4.4 Further Discussions

Overall, our Comparison studies show that the cloud gaming systems increases the CPU consumption and as well as power consumption. In addition to this, virtualization consumes a significant amount of CPU and as well as a large amount of RAM. Our experiments also indicate that the performance of cloud gaming system decreases after virtualization drastically. Although this performance drop depends on the type of virtualization and the ability of virtualization software used, our results present a general case and support our idea that the performance of games running on a cloud gaming system inside a VM is inferior as compared to the same game running on a bare metal computer.

# 5 Optimization Of Cloud gaming Systems

In this section, we propose an approach to enhance the performance of cloud gaming system and present our results. Our initial experiments show that a cloud gaming system increases the latency in games and also introduces a significant amount of overhead in consumption of various resources over and above what a game requires.

This is because, in a cloud gaming system, the server receives the gaming inputs in the form of packets over the network on the physical NIC(Network Interface Controller) in turn pass them on to virtual NIC of the VM on which the game is running. These packets arriving on Physical NIC generate hardware interrupts and the packets transferred to the virtual NIC on VM generates software interrupts. The cloud gaming system on the VM, in turn, decodes these packets to actual inputs and passes it to the game. The game engine uses these inputs to render the game scenes and generates the game output in the form of game video and Audio and passes it to the Operating system. The operating system stores these frames in back buffers before transferring them to the front buffer for display. Then the cloud gaming system captures these frames either from the front buffer or back buffer of the operating system depending on the implementation. These frames are compressed and encoded typically using an encoder like H.264/MPEG-4 AVC. The compressed video is broken into packets and these packets are forwarded to the client using the virtual NIC of VM and in turn using the physical NIC of the server. Most of these operations require a lot of read

and write operations done on the memory. The additional steps required in a virtual machine over its non-virtual counter part while performing these operations are the main source of increase in the CPU as well as power consumption.

Our intuition is that if the host OS and the guest VM share memory space, there will be a decrease in the number of interrupts generated and the number of CPU cycles required to copy data between the host and the guest as there will be a zero copy between the host and guest system. As a by-product, there will also be a decrease in the energy consumption as a result of the shared memory [5].

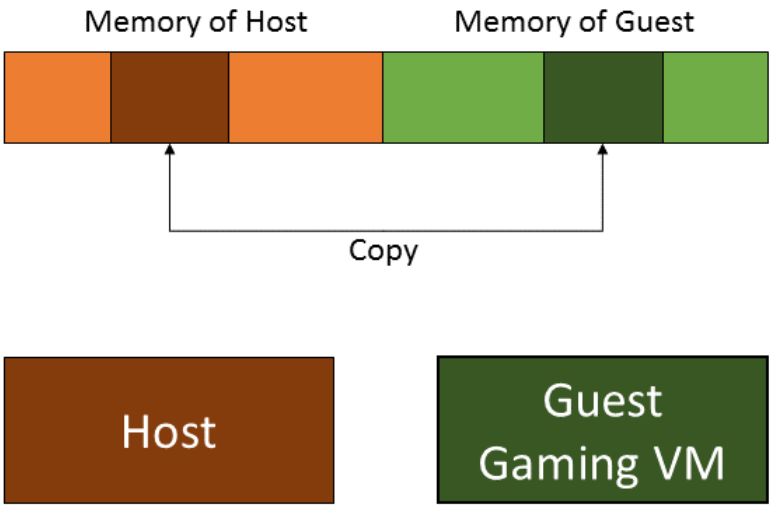


Figure 5.1: Host And Guest Memory Copy

## 5.1 Shared Memory Implementation

In this section, we present the techniques we used to establish a shared memory between host Operating System and the guest VM.

We setup an environment using Cloudstack for our experiments. Cloudstack[3] is an open source cloud computing software that can be used to create, deploy and manage cloud services. We use KVM(Kernel Virtual Machine)[1] for our VM deployments. KVM provides type 1 or hardware virtualization. This means that a KVM hypervisor directly resides over the hardware and provides hardware access to the guest OS through it without the need for a host OS. We use a 9p over virtio along with file system device for establishing a shared memory. 9p-virtIO(Plan 9 folder sharing over VirtIO) is a tool that KVM provides. Instead of providing a virtual memory to the guest VM, a paravirtual file system driver is used which provides the guest VM access to storage on the host. This mechanism avoids conversion from guest application file system operations into block device operations and in turn into host file system operations.

Firstly, we setup a QEMU(Quick Emulator) server on the host and export a portion of the file system on the host. Creating shared space using QEMU is natively supported on Linux operating system. This is done using the following command.

```
1 # /usr/bin/qemu-kvm -m 1024 -name centos6.2 local, \
2 security_model=passthrough, id=fsdev0, \
3 path=/share -device virtio-9p-pci, \
4 id=fs0, fsdev=fsdev0, mount_tag=hostshare
```

In the above command, mount_tag is the unique identifier for this shared memory.

The client for QEMU server is used on guest VM. This client is used to mount the file system exported by QEMU server on the guest like a local file system. This can be done by using the mount_tag as follows:

```
1 # mkdir tmp/hostFiles
```

[1] https://www.linux-kvm.org/

```
2 # mount -t 9p -o trans=virtio, \
3 version=9p2000.L hostshare tmp/hostFiles
```

We initially create a new directory called hostFiles and then mount the shared file system from the host using the mount command. The guest file system is free to perform read/write operations on this shared space as it does on any local directory. The guest sees this as any other file system while the actual read/writes that are done on this actually happen on the host file system. This allows a shared space in such a way that both host and the guest achieve a zero-copy. We setup our gaming system such that all I/O operations happen in this shared space. This ultimately eliminates the need for data copy between the host's memory and guest's memory. The communication between the host and guest happen using the plan-9 network protocol used by 9p-virtio.

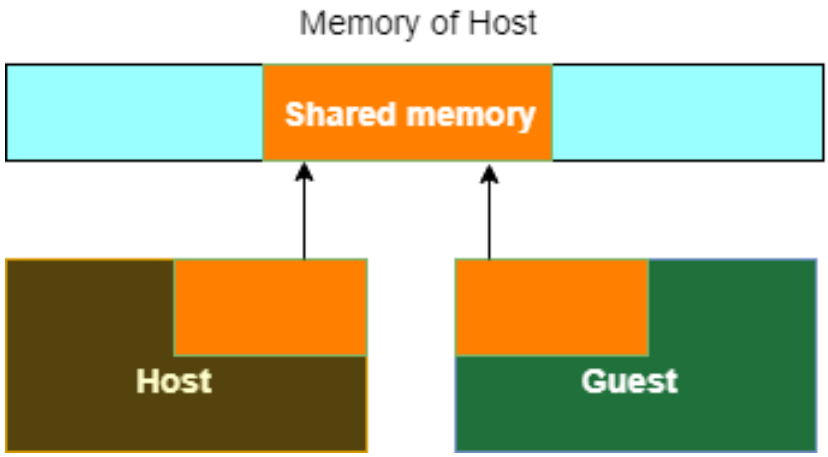


Figure 5.2: Memory Sharing Architecture

## 5.2 Evaluation

In this section, we present the evaluation of the proposed approach. We run the same games from our previous experiments except Asphalt 8: Airborne as it doesn't have a Linux version. We compare the performance of cloud gaming system on a VM without memory sharing implemented with its memory sharing enabled counterpart. We measure the latency of the game and the frame rate at which the game runs on the client.

The results for latency measurements were as follows:

| Game | Without Memory sharing | With Memory Sharing |
|---|---|---|
| **Red Eclipse** | 513.23 ms | 475.70 ms |
| **0 A.D.** | 590.15 ms | 547.93 |

Figure 5.3 shows the latency comparison results of our experiments.

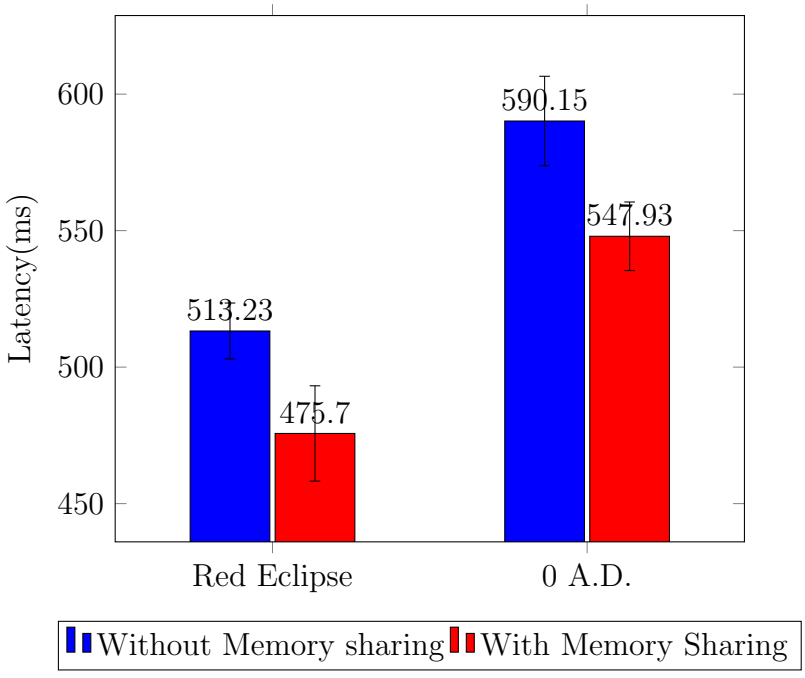


Figure 5.3: Latency Comparison

Latency decreases by 7.3% in Red Eclipse with memory sharing enabled on the VM. In case of 0 A.D., this decrease is about 7.1%. This results show that there is an average improvement of 7.2% in latency as a result of memory sharing.

The results for frame rate were as follows:

| **Game** | **Without Memory sharing** | **With Memory Sharing** |
|---|---|---|
| **Red Eclipse** | 7.5 fps | 7.98 fps |
| **0 A.D.** | 8.02 fps | 8.48 fps |

Figure 5.4 shows the latency comparison results of our experiments.

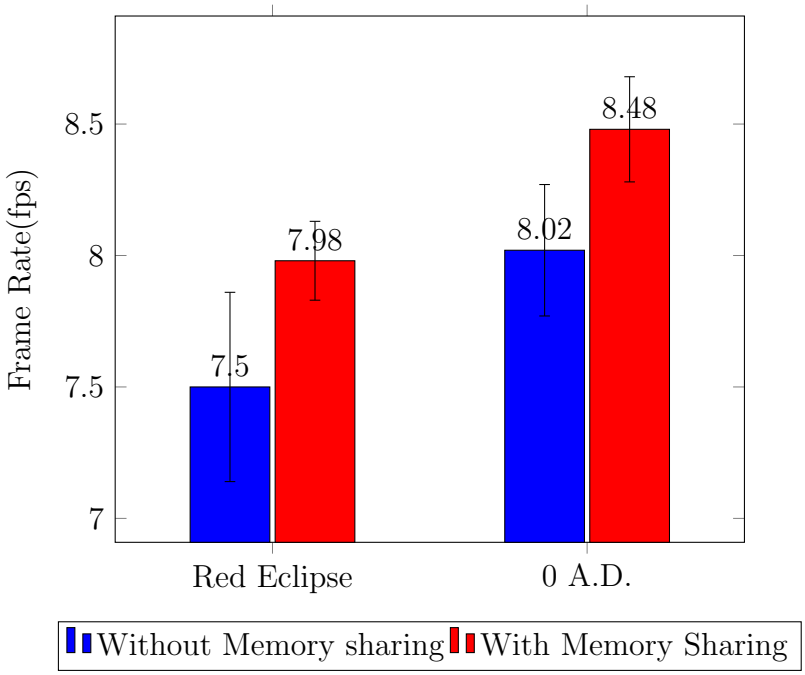


Figure 5.4: Frame Rate Comparison

Frame rate increases by 6.9% in Red Eclipse with memory sharing enabled on the VM. In case of 0 A.D., this improvement is about 6.2%. Here we observe that the average frame rate increases by about 6.55% due to memory sharing.

When memory is shared between guest VM and host OS, we observe that the performance of cloud gaming system improves. The decrease in latency can be attributed to the faster memory copy operations as there is no need to copy data from host OS to guest VM and then back to the host OS again ultimately resulting in lower number of interrupts. This translates to saved time and hence a decrease in latency. Also, this allows the OS to allocate more CPU cycles in a given amount of time which would have otherwise been used for these activities resulting in improved frame rate. Overall, we observe a decrease of 7.2% in latency and an increase of 6.05% in frame rate on an average.

# 6 Conclusions

Our work is aimed at understanding the performance of cloud gaming systems on virtualized environments. For this, we initially measure the performance of cloud gaming systems in various environments. Initially, we run a chosen set of 3 games belonging to various genres locally on a computer. Then, we stream the same game to a client using the open-source cloud gaming system Gaming Anywhere. Later, we repeat the same experiment inside a VM. We measure the performance of cloud gaming system and the resource consumption in each of these scenarios. We measure various factors like latency, frame rate, CPU consumption, RAM usage and power utilization. Our experiments show that a cloud gaming system needs a lot of additional resources compared to a bare metal system running the same game. To improve the performance, we implement a memory sharing mechanism between the host and guest VM using 9p-virtio. This creates a shared memory space on the host that a guest can mount so the guest VM can read and write in this shared space. We observe that our setup shows an improvement of 6% in frame rate, a decrease of 7.2% in latency.

# Bibliography


[1] *0 A.D. Website*. URL: https://www.play0ad.com/ (visited on 06/12/2017) (cit. on p. 21).

[2] M. Ahmed, A. Chowdhury, M. Ahmed, and M. M. H. Rafee. "An advanced survey on cloud computing and state-of-the-art research issues". In: *IJCSI International Journal of Computer Science Issues* 9.1 (2012), pp. 1694–0814 (cit. on p. 7).

[3] *Apache Cloudstack*. URL: https://cloudstack.apache.org (cit. on p. 44).

[4] *Asphalt 8*. URL: https://www.gameloft.com/asphalt8/ (visited on 06/12/2017) (cit. on p. 20).

[5] Y. Bai. *Power Consumption of Virtual Machines in Cloud Computing: Measurement and Enhancement*. 2015 (cit. on p. 43).

[6] D. C. Barboza, H. L. Junior, E. W. G. Clua, and V. E. F. Rebello. "A Simple Architecture for Digital Games on Demand Using Low Performance Resources under a Cloud Computing Paradigm". In: *2010 Brazilian Symposium on Games and Digital Entertainment*. Nov. 2010, pp. 33–39. DOI: 10.1109/SBGAMES.2010.34 (cit. on p. 15).

[7] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei. "Measuring the Latency of Cloud Gaming Systems". In: *Proceedings of the 19th ACM*

*International Conference on Multimedia*. MM '11. Scottsdale, Arizona, USA: ACM, 2011, pp. 1269–1272. ISBN: 978-1-4503-0616-4. DOI: 10.1145/2072298.2071991. URL: http://doi.acm.org/10.1145/2072298.2071991 (cit. on p. 23).

[8] K.-T. Chen, C.-Y. Huang, and C.-H. Hsu. "Cloud gaming onward: research opportunities and outlook". In: *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*. July 2014, pp. 1–4. DOI: 10.1109/ICMEW.2014.6890683 (cit. on pp. 12, 14).

[9] M. Claypool and D. Finkel. "The effects of latency on player performance in cloud-based games". In: *Network and Systems Support for Games (NetGames), 2014 13th Annual Workshop on*. Dec. 2014, pp. 1–6. DOI: 10.1109/NetGames.2014.7008964 (cit. on p. 13).

[10] C. Ding, Y. Chen, T. Xu, and X. Fu. "CloudGPS: A Scalable and ISP-friendly Server Selection Scheme in Cloud Computing Environments". In: *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*. IWQoS '12. Coimbra, Portugal: IEEE Press, 2012, 5:1–5:9. ISBN: 978-1-4673-1298-1. URL: http://dl.acm.org/citation.cfm?id=2330748.2330753 (cit. on p. 16).

[11] M. Emma. *THE GLOBAL GAMES MARKET WILL REACH $108.9 BILLION IN 2017 WITH MOBILE TAKING 42%*. URL: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42 (visited on 05/07/2017) (cit. on pp. 3, 6).

[12] S. Hollister. *OnLive lost: how the paradise of streaming games was undone by one man's ego*. 2015. URL: http://www.theverge.com/2012/8/28/3274739/onlive-report (visited on 12/18/2015) (cit. on p. 4).

[13] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. "Placing Virtual Machines to Optimize Cloud Gaming Experience". In: *Cloud Computing, IEEE Transactions on* 3.1 (Jan. 2015), pp. 42–53. ISSN: 2168-7161. DOI: 10.1109/TCC.2014.2338295 (cit. on p. 14).

[14] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen. "GamingAnywhere: An Open Cloud Gaming System". In: *Proceedings of the 4th ACM Multimedia Systems Conference*. MMSys '13. Oslo, Norway: ACM, 2013, pp. 36–47. ISBN: 978-1-4503-1894-5. DOI: 10.1145/2483977.2483981. URL: http://doi.acm.org/10.1145/2483977.2483981 (cit. on pp. 13, 16).

[15] A. Jurgelionis, F. Bellotti, A. D. Gloria, J. P. Laulajainen, P. Fechteler, P. Eisert, and H. David. "Testing Cross-Platform Streaming of Video Games over Wired and Wireless LANs". In: *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*. Apr. 2010, pp. 1053–1058. DOI: 10.1109/WAINA.2010.186 (cit. on p. 16).

[16] D. Klionsky, A. Treuille, and S. Narasimhan. *A New Architecture for Cloud Rendering and Amortized Graphics*. 2011 (cit. on p. 15).

[17] U. Lampe, Q. Wu, S. Dargutev, R. Hans, A. Miede, and R. Steinmetz. "Assessing latency in cloud gaming". In: *International Conference on Cloud Computing and Services Science*. Springer. 2013, pp. 52–68 (cit. on p. 24).

[18] *Liquid Sky Website*. URL: https://www.liquidsky.com/ (visited on 07/01/2017) (cit. on p. 16).

[19] X. Lou and K. Hwang. "Quality of Data Delivery in Peer-to-peer Video Streaming". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 8.1S (Feb. 2012), 12:1–12:23. ISSN: 1551-6857. DOI: 10.1145/2089085.2089089. URL: http://doi.acm.org/10.1145/2089085.2089089 (cit. on p. 15).

[20] X. Meng, V. Pappas, and L. Zhang. "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement". In: *Proceedings of the 29th Conference on Information Communications*. INFOCOM'10. San Diego, California, USA: IEEE Press, 2010, pp. 1154–1162. ISBN: 978-1-4244-5836-3. URL: http://dl.acm.org/citation.cfm?id=1833515.1833690 (cit. on p. 3).

[21] *Parsec Website*. URL: https://www.parsec.tv/ (visited on 07/01/2017) (cit. on p. 16).

[22] L. Rao, X. Liu, L. Xie, and W. Liu. "Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment". In: *2010 Proceedings IEEE INFOCOM*. Mar. 2010, pp. 1–9. DOI: 10.1109/INFCOM.2010.5461933 (cit. on p. 3).

[23] D. Rapp. *The Mother of All Video Games*. Nov. 2006. URL: https://web.archive.org/web/20080517011435/http://www.americanheritage.com/people/articles/web/20061129-pong-video-games-nolan-bushnell-atari-al-alcorn-nintendo.shtml (cit. on p. 5).

[24] *Red Eclipse*. URL: https://www.redeclipse.net/ (visited on 06/12/2017) (cit. on p. 20).

[25] R. Shea, J. Liu, E. Ngai, and Y. Cui. "Cloud gaming: architecture and performance". In: *Network, IEEE* 27.4 (2013), pp. 16–21 (cit. on pp. 6, 12).

[26] R. Süselbeck, G. Schiele, and C. Becker. "Peer-to-peer support for low-latency Massively Multiplayer Online Games in the cloud". In: *2009 8th Annual Workshop on Network and Systems Support for Games (NetGames)*. Nov. 2009, pp. 1–2. DOI: 10.1109/NETGAMES.2009.5446229 (cit. on p. 16).

[27] *Vortex Website*. URL: https://www.vortex.gg/ (visited on 07/01/2017) (cit. on p. 16).

[28] Z. Xue, D. Wu, J. He, X. Hei, and Y. Liu. "Playing High-End Video Games in the Cloud: A Measurement Study". In: *Circuits and Systems for Video Technology, IEEE Transactions on* PP.99 (2014), pp. 1–1. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2014.2364419 (cit. on p. 13).

[29] Z. Zhao, K. Hwang, and J. Villeta. "Game Cloud Design with Virtualized CPU/GPU Servers and Initial Performance Results". In: *Proceedings of the 3rd Workshop on Scientific Cloud Computing Date*. ScienceCloud '12. Delft, The Netherlands: ACM, 2012, pp. 23–30. ISBN: 978-1-4503-1340-7. DOI: 10.1145/2287036.2287042. URL: http://doi.acm.org/10.1145/2287036.2287042 (cit. on p. 14).